XRDirector: A Role-Based Collaborative Immersive Authoring System

Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, Janet Nebeling
University of Michigan School of Information
nebeling@umich.edu — https://mi2lab.com

ABSTRACT
Immersive authoring is an increasingly popular technique to design AR/VR scenes because design and testing can be done concurrently. Most existing systems, however, are single-user and limited to *either* AR *or* VR, thus constrained in the interaction techniques. We present *XRDirector*, a role-based collaborative immersive authoring system that enables designers to freely express interactions using AR *and* VR devices as puppets to manipulate virtual objects in 3D physical space. In XRDirector, we adapt roles known from filmmaking to structure the authoring process and help coordinate multiple designers in immersive authoring tasks. We study how novice AR/VR creators can take advantage of the roles and modes in XRDirector to prototype complex scenes with animated 3D characters, light effects, and camera movements, and also simulate interactive system behavior in a Wizard of Oz style. XRDirector's design was informed by case studies around complex 3D movie scenes and AR/VR games, as well as workshops with novice AR/VR creators. We show that XRDirector makes it easier and faster to create AR/VR scenes without the need for coding, characterize the issues in coordinating designers between AR and VR, and identify the strengths and weaknesses of each role and mode to mitigate the issues.

Author Keywords
AR/VR; immersive authoring; mixed-reality collaboration.

CCS Concepts
•**Human-centered computing** → **Mixed / augmented reality;** *Collaborative interaction;*

INTRODUCTION
Immersive authoring enables asset creation and programming of interactive behavior from within the virtual environment [20]. It is a technique that allows verifying user perception of AR/VR content and interaction usability *in situ* [14, 35]. Many key vendors have an immersive authoring system (Google Tilt Brush, Oculus Quill, Microsoft Maquette, Disney's PoseVR, etc.). While they are becoming increasingly powerful and popular (e.g., Google's artist in residence program), they all are designed for single-user usage and typically attempt to move the entire authoring system into *either* AR *or* VR.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI'20, April 25–30, 2020, Honolulu, HI, USA

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-6708-0/20/04...$15.00
DOI: https://doi.org/10.1145/3313831.3376637

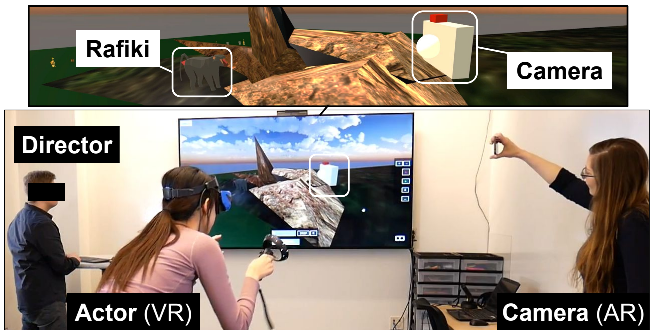

Figure 1. ***XRDirector*** **enables designers in different roles using mixed-reality devices to jointly enact prototypes using AR/VR interactions. In our recreation of the Lion King (2019) teaser, the *director* instructed an *actor* playing the Rafiki character to move up Pride Rock in VR and a designer controlling the *camera* to focus on Rafiki in AR.**

The goal of our work is to study the issues in enabling multiple designers to collaborate in immersive authoring tasks, taking advantage of *both* AR *and* VR where appropriate. XRDirector targets co-located settings in which multiple designers collaborate at the same time and in the same place [11], which provides many new challenges. Prior work addressed time-based and, to a lesser extent, spatial coordination issues, but only considered scenarios where the entire team uses AR [3, 18, 9, 16] or only one user is in VR [4, 6]. Our work identifies new, interesting issues as some designers work in VR and others in AR. We group these issues into four main categories: *spatial awareness*, *constraints*, *mappings*, and *resolutions*.

To study these issues, we present *XRDirector*, a multi-user, cross-device, immersive authoring system (Figure 1). XRDirector demonstrates a role-based collaborative authoring approach that adapts roles known from filmmaking to structure authoring tasks and help coordinate multiple designers. It is through the roles combined with specific AR/VR modes adapting view and behavior that we are able to address most coordination issues in XRDirector. We aimed to give each role in XRDirector autonomy in control and movement over certain elements of a scene, but limited the number of aspects each individual user needs to pay attention to while increasing the need for collaboration. Traditional filmmaking is focused on executing the director's vision, involving coordination of all moving parts, instruction to everyone on set, and verification of all takes. Since the director is crucial to the production's success, we developed a special role with full control over a scene, including placement and movement of 3D characters, lights, and the camera, even while being embodied and controlled by other designers in our XRDirector system.

XRDirector was designed to study how novice AR/VR creators can be empowered to create complex scenes, with multiple animated 3D characters, light effects, and camera movements, and to simulate interactive behavior in a Wizard of Oz style. Our primary contribution is a set of immersive authoring techniques adapted from filmmaking, as well as showing how they can be leveraged to structure collaborative authoring. We also see value in XRDirector as a system for early-stage design explorations, particularly when there is no animation script or interaction design concept yet, but it is not meant to substitute for more advanced AR/VR tools [28]. XRDirector incorporates principles from layered 3D character animation [8] and 3D puppetry [1, 2, 15], but we take the crucial next step of extending it to collaborative authoring in mixed reality [33].

We report on our design process around XRDirector in three steps. First, we conducted three case studies around complex 3D movie scenes and AR/VR games, initially recreating them using low-fi prototyping without technical constraints, then using digital tools and scripting, to better understand requirements and inform the design of XRDirector. Second, we conducted workshops with novice AR/VR developers, discussing their major challenges with existing tools, exploring the potential and limitations of XRDirector when using it to recreate selected scenes. Third, we characterized the issues when reenacting the prototype scenes from our case studies using XRDirector, and explored additional applications to assess its expressive power and flexibility.

BACKGROUND

XRDirector adds to prior AR/VR prototyping and Wizard of Oz tools, and performance animation and puppetry interfaces.

AR/VR Prototyping and Wizard of Oz Tools

Early immersive authoring tools allowing designers to create AR/VR content directly in AR/VR were inspirational to our work (e.g., [14, 20, 35]. Our work also shares goals with Studierstube [31] and DART [24, 13], with initial support for collaboration and Wizard of Oz based on record & replay of sensor data [9, 10]. More recent tools like ARcadia [17], ProtoAR [27], and 360proto [26] focus on creating mappings between physical artifacts and digital representations for AR/VR—ARcadia adapts a block-based scripting language for AR and allows markers to function as input and control; ProtoAR generates mobile screens with embedded AR scenes from paper sketches and Play-Doh models; 360proto turns paper sketches themselves into 360-degree representations of AR/VR environments. XRDirector is a continuation of these tools: it captures physical movement and translates it into 3D animations of virtual objects during immersive authoring.

Performance Animation and Puppetry Interfaces

We think of XRDirector as an extension of earlier performance animation tools. Dontcheva et al. [8] used a multi-camera motion capture system to track an actor for 3D character animation. Video and 3D puppetry [2, 15] are camera-based systems tracking paper cut-outs and physical objects moved by a puppeteer for 3D animation. Cellphone Puppets [29] and PuppetPhone [1] directly turned AR phones into puppets. XRDirector extends these techniques to multi-user collaboration scenarios. While we can leverage inside-out tracking of devices without building special-purpose pipelines, our main issue becomes coordinating multiple designers in AR/VR.

This was previously shown to require careful timing and extensive training [16, 18]. Haptic Turk used preemptive warnings [4] and Turk Deck [6] a scheduler with delay mechanism to leverage humans as actuators to create a realistic, immersive experience. These systems required additional devices such as iPads and a projector for coordination displays. Without turn-taking techniques [5] they are limited to one VR player. With XRDirector, we present a role-based framework for coordinating co-located teams of designers in mixed reality, without extensive instrumentation or additional displays.

Other related systems include GestureWiz [34] and Montage [22]. GestureWiz uses human wizards to simulate gesture-based interfaces. Montage provides a comprehensive video prototyping suite to simulate interactive systems with support for AR. We developed XRDirector in the same spirit, but wanted to produce system output beyond videos so that the resulting prototypes do not just live or die with the prototyping tool. We share the goals of sketch-based animation tools, K-Sketch [7], Apparition [19], and SketchExpress [21], in that we also want to generate complex animations from simple demonstrations. We present a solution to generate 3D animations and interactive behavior from demonstrated AR/VR interactions in XRDirector. Similar to [8, 23], XRDirector implements post-production tools to transform tracking data along both the time and space dimensions. Unlike prior work, we study how these techniques can be used to overcome spatial coordination challenges when coordinating multiple designers.

WORKSHOPS WITH NOVICE AR/VR DEVELOPERS

In the following, we describe our process of learning about requirements for new tools like XRDirector. In a first step, we wanted to gather feedback from novice AR/VR developers regarding their experience with existing tools and challenges they commonly faced when creating AR/VR prototypes. In a second step, we carried out three case studies recreating existing 3D movie and AR/VR games scenes.

Structure. We recruited 12 master's students (10 females, 2 males, with a median age of 25) from an AR/VR development course with a median of 3 months experience working with A-Frame and Unity. We started the workshops by asking the students to present a challenging AR or VR scene they had developed. We asked each student to walk us through key features of their scene and explain the major challenges they had creating the scene. After all students had stated their challenges, we asked them each to pick one main challenge.

Findings. In total, we saw a good mix of AR/VR scenes. To create these scenes, 7 students used A-Frame and 5 students used Unity. Each of the scenes presented had an average of 2–3 different types of animations (scripted 3D character motions, light effects, camera movements, particle systems). Overall, students found difficulty with animating objects in their scenes in a natural way. In particular, they found it tedious to create lots of concurrent animations and hard to do this efficiently. In this same vein, they found it difficult to apply unique motions to objects, as this required manually coding each individual movement. While some issues may be resolved with more experience in AR/VR programming or using additional animation tools, our students are not the first to report such issues and to see them as major barriers [2, 15].

CASE STUDIES

We conducted three case studies around complex 3D movie scenes and AR/VR games, each involving two rounds of prototyping: *1)* physical prototyping with no technical considerations, *2)* digital prototyping in existing tools to create and experience a prototype on AR/VR devices. We used common tools for 3D modeling and immersive authoring (Blender and Google Blocks) and AR/VR platforms (A-Frame and Unity).

Case Study 1: Lion King (2019) Teaser

As our first case study, we recreated elements from the first teaser trailer of the Lion King (2019) remake. We selected it because of its high quality animations. It was also an interesting case because filming was done completely in VR [12].

Physical Prototype

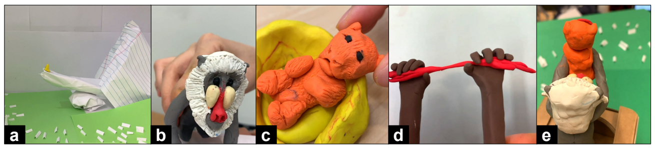

Figure 2. Physical prototype based on Lion King teaser to understand requirements for camera and 3D character movements.

Figure 2 shows a collage of different elements we recreated using paper prototyping in combination with Play-Doh based on the methods presented in [27]. From the physical prototype, we learned how a variety of different camera angles and movements can come together in a scene, from a far away view of Pride Rock, to close-ups of Simba (Figure 2a and c). We also learned how differently detailed models were required for the same 3D character to accomplish different types of motions. For example, in one scene, Rafiki moves from climbing up a rock to breaking a vine over his head with his arms (Figure 2b and d). It was easier to create two models for those visuals than to try to contort one model into both positions.

Digital Prototype

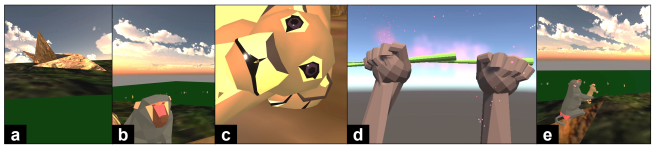

Figure 3. Digital Lion King recreation with 3D models created in Blocks and Blender as well as scripted animations in A-Frame and Unity.

While creating the digital prototype of the Lion King scenes, we noticed the complexity in terms of 3D character animation, camera movements, and scene transitions (Figure 3). While we were able to replicate most camera movements and compositions with reasonable accuracy, our scenes did not match the quality of the original trailer. To achieve higher fidelity, our prototype would need to have much more detailed models, with hair, fur and movable joints, which would have required many weeks of model sculpting, rigging, and animation in several 3D tools. Using our selected tools to create the scenes was tedious, as moving and placing objects as well as scripting object movements required a lot of trial and error.

Case Study 2: Beat Saber

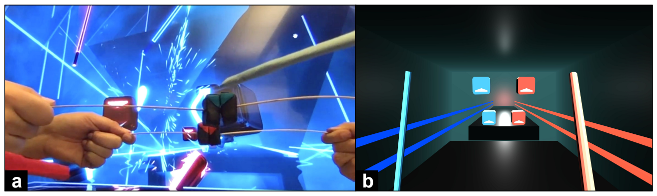

Figure 4. Physical and digital recreations of Beat Saber to study requirements for rapid object movement, lighting, and sound in VR.

Our second case study was a recreation of the VR game, Beat Saber (Figure 4). In the game, blocks move towards the player who swings the sabers to slash the blocks in half. We chose this case because of its relative popularity and the rapid object movements coordinated with light and sound effects.

Physical Prototype

Even though the video game displays a seemingly simple interface with straightforward interactions, it was challenging to recreate the steady and measured movements of the game blocks. In the physical prototype, we were able to reproduce all of the scene elements but compromised on the flashing background lights. Cutting the block in half was also difficult to simulate, requiring coordinated movement between the player and two puppeteers, each holding a piece in both hands.

Digital Prototype

With our digital prototype of Beat Saber, we found that adjusting the light to match the beat of the music required a lot of manual scripting and tuning of animation timings. Though we simplified the background and some effects from the original, such as the cubes not splitting in half, we were able to recreate the basic game mechanics with few challenges in A-Frame.

Case Study 3: ARcade Plane

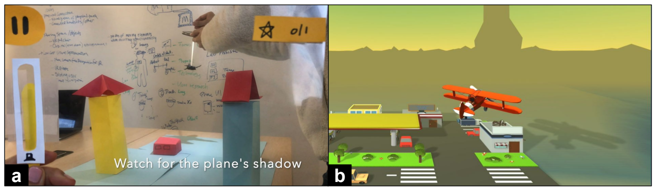

Figure 5. Recreations of ARcade Plane to learn about requirements for 3D character movement, directional light, and shadows in AR.

The third case study was a replication of the AR game, ARcade Plane (Figure 5). In the game, the user is given a task to navigate a plane around objects in a digital town. The plane is controlled by moving and tilting the phone, a common set of mobile AR interactions, which is why we chose this case.

	Viewing Scene				Authoring Scene						Post-Processing		
Functions / Roles	View Avatars	View 3D Characters	View Lights	View Camera	Control 3D Characters	Control Lighting	Control Camera	Scale Scene	Record Animation	Replay Animation	Generate Animation Script	Scale Animation Trajectory	Layer Animation Sequences
Director	X	X	X	X	X	X	X	X	X	X	X	X	X
Actors	X	X	O	X	X	-	-	O	-	O	-	O	O
Lights	X	X	X	X	-	X	-	O	-	O	-	O	O
Camera	X	X	O	X	-		X	X	X	X	-	O	O
Viewer	-	X	O	O	-	-	-	-	-	O	-	O	O

Figure 6. Roles and functions in XRDirector: X = role can execute function; O = role sees result of function; – = role does not support function.

Physical Prototype

We created the 2D HUD using transparency and 3D characters out of paper and Play-Doh, replicating most aspects of the original game but not including sound. One puppeteer moved the plane model and also used their phone's flashlight for lighting. Close coordination between the puppeteer and the player was required, since the puppeteer needed to guess the player's intended actions. This required verbal communication and a shared sense of the anticipated end result.

Digital Prototype

Our digital prototype focused on the 3D characters from the original game—replicating the 2D HUD was secondary. For the plane and light movements, each motion step needed to be scripted in terms of position and rotation changes, which required many lines of code for only a small set of movements.

REQUIREMENTS

From these case studies, we identified three main requirements that we wanted to address with XRDirector:

R1: Flexible workflows for prototyping 3D animations

Physical prototyping was easy and highly iterative. After a first recording was taken, missing parts could be added, existing ones remolded, and the scene shot again until satisfied. On the other hand, if only some elements of an animation did not feel right, typically the entire animation had to be recreated, causing another iteration. Digital prototyping required careful planning and felt much more constrained. 3D animation required a lot more effort and was tedious with scripting. Recording was typically delayed until all scene elements were in place. In XRDirector, we aimed to enable flexible workflows similar to physical prototyping, by allowing animations to be demonstrated without scripting, layered to support iterations, and modified to make corrections post-hoc.

R2: Joint control over multiple moving parts

Physical prototyping afforded creative use of additional materials to make physical props stick and move with a person or other objects. We felt that we achieved better results the more control we could exercise over the object, which was hard for particularly small or large objects. Often it was best to take objects apart and allow individual parts to be controlled by more than one person or using additional people to fix parts that should not be moving. We used the same strategy in digital prototyping though it was more tedious to produce complex animations from multiple moving parts via scripting. In XRDirector, we wanted to enable designers to jointly create complex scenes by embodying multiple 3D characters or assisting with detailed animations of the same character.

R3: Clearly established roles to facilitate collaboration

Physical prototyping made it easy to lend each other a hand and help with filming while 3D characters were being animated in front of the camera. While roles stayed mostly the same, the camera operator often also worked as director by quickly adding or removing elements from the scene. Collaboration on digital prototypes was much more difficult to coordinate despite using online editors such as CodePen and Glitch with state-of-the-art collaboration tools. In XRDirector, we wanted to facilitate collaboration between designers by establishing clearly defined roles. Initially, we only distinguished between the director and puppeteer roles, but after our case studies, we strongly felt the need for a separation of roles into actors, lights, and the camera, giving full control over all scene elements only to a user in the director role.

XRDIRECTOR

At the highest level, XRDirector distinguishes two main concepts: *roles* and *modes*. Roles are assigned to users and determine *what* types of virtual objects in the scene users can view and manipulate. Modes are assigned to devices and determine *how* users view and author the scene.

Roles

We developed five roles in XRDirector (Figure 6):

- **actors** animate 3D characters and produce spatial sound;
- **lights** control lighting of the scene to create basic effects;
- **camera** records 3D character animations and light effects;
- **director** orchestrates actors, lights, and the camera via verbal instructions, or takes direct control over them (e.g., to move them and demonstrate a performance);
- **viewers** can see the live performance or watch a recording.

As shown in Figure 6, the director has full control over the entire system, while other roles only have access to a subset of the system functions. This effectively enables usage by a single designer in the role of director or focusing on the particular requirements of 3D character animation, light effects, and camera transitions when multiple designers are involved.

We designed our role-based framework that way to reduce the perceived complexity, hence the learning curve, of our system, and enhance collaboration, hoping that it would help distribute the work between designers and make it easier for the director to instruct designers on a per-role basis. While roles other than the director have specific, fixed functions, switching roles only requires changing a parameter in XRDirector. This enables quick transition to other roles in our collaboration framework.

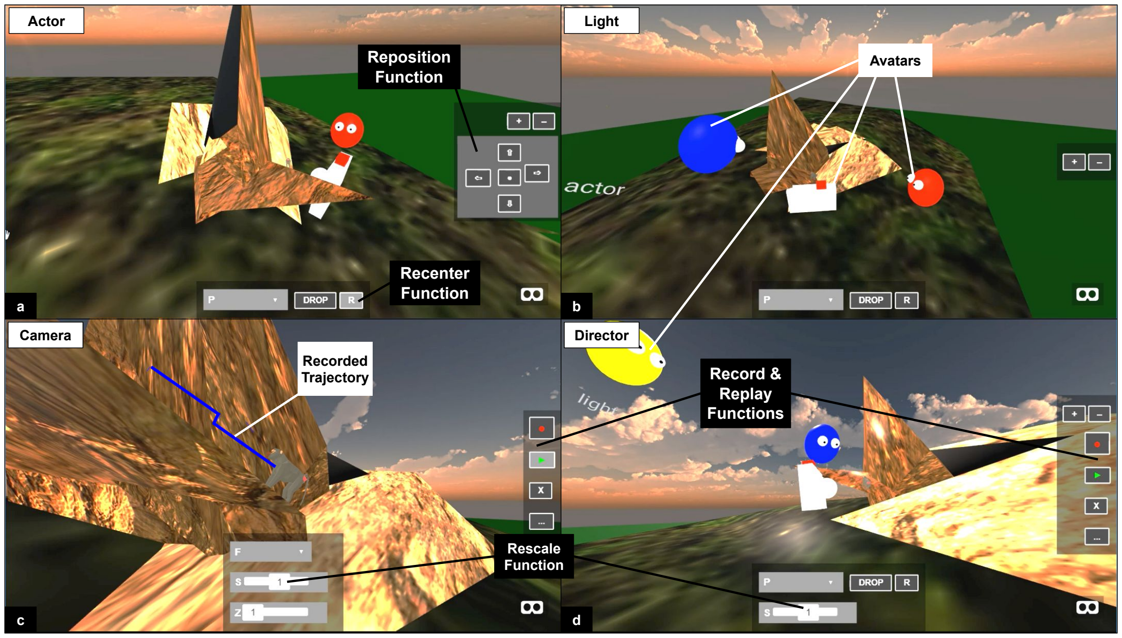

Figure 7. Recreation of the Lion King Teaser in XRDirector showing the views of four designers in different roles side-by-side: *actor*, *light*, *camera*, *director* (all represented by avatars): *a)* the actor moves the Rafiki character up Pride Rock; *b)* the *light* dims the ambient light to crossfade between scenes; *c)* the camera zooms and keeps Rafiki in view; and *d)* the director oversees the scene, instructs the actor, light, and camera, and functions as their common reference point. Widgets (in gray) floating over the scene can be used to edit selected virtual objects and perform role-specific functions.

To facilitate coordination in embodied mixed reality [32, 33], all roles are represented by virtual avatars (Figure 7b). The camera is a fixed 3D model with an enlarged lens; other roles are color-coded heads with enlarged eyeballs. The lens and eyes show orientation and indicate each role's perspective on the scene. When actors and lights select a 3D character or light source in the scene, their avatars are hidden such that they effectively become the selected virtual object.

Viewers only see the 3D characters and light sources, not the actors' virtual avatars. This makes for a clean viewer experience, gives the illusion of a "working prototype" in Wizard of Oz applications of XRDirector, and addresses a limitation of previous works that struggled to remove the puppeteer or wizard from viewers [2, 26]. Multiple viewers are possible and share the view of the camera, as in the movies, or move freely in the scene, making it interactive.

The camera and director roles have access to a special **scale function** to resize all virtual objects at once while keeping their position and size in the scene fixed. We added this feature to better align the virtual world with the physical world and space available to designers to express interactions. Note that this scale function does not take effect for viewers who always view the scene at true scale. We will later demonstrate the importance of this function for spatial coordination with multiple designers using XRDirector in different modes.

Modes

We also developed three modes in XRDirector:

- **Plain 3D mode**: the user views a perspective rendering of the scene on a computer screen and transforms the view and a selected object by using WASD keys and dragging the mouse similar to FPS games. It is the basis of other modes.
- **AR mode**: the user views the scene as magic window with skybox or rear camera feed on a hand-held or head-worn AR device and transforms the camera perspective and a selected object by moving the device in physical space.
- **VR mode**: the user views a stereoscopic rendering of the scene on a VR headset and transforms the view and a selected object by moving the headset in physical space.

Plain 3D is similar to common 3D digital authoring tools, while AR/VR modes enable immersive authoring.

For the VR mode, we added an option to attach the selected object to the controller to move it separately from the designer's view of the scene. We did this to match the flexibility of the AR mode supporting hand-held and head-worn devices. It was also motivated by our case studies where we varied the body parts to better fit requirements of natural movements.

In addition to a **teleport function** common to many AR/VR applications for travelling virtual distance without moving physically, each mode implements two functions to deal with calibration and locomotion differences between modes: a **recenter function** moves the virtual world so that it is centered on the designer's virtual position independent of their physical location; a **reposition function** moves the selected object relative to the designer's virtual position or relative to the VR controller's position if that option is selected.

The first function allows a designer to realign the virtual world around their position in physical space. This enables calibration with collaborators without having to restart the system and also helps recover from AR/VR tracking issues, which still frequently occur with inside-out tracked devices.

The second function allows a designer to keep the virtual object in view while they are manipulating it as they would otherwise occupy the same virtual space as the object they are controlling. This can be crucial for a designer when getting familiar with a new role. Seeing the character movement in third person can also enhance feedback and feel of control.

USING XRDIRECTOR

Figure 7 shows our Lion King recreation detailed later with one designer controlling lighting in AR mode to lit the scene, a second designer moving the camera in AR mode to keep Rafiki in frame, a third controlling the Rafiki character in VR mode, and the director coordinating all three in the plain 3D mode. Note that changes made to the scene by one designer are immediately propagated to designers on other devices.

Lights, Camera, Action!

Below we walk through the experience of using XRDirector to prototype scenes from our Lion King case study. While we keep the examples relatively simple, the scenes have sufficient complexity to demonstrate key aspects of XRDirector.

Controlling Lights

A key aspect when authoring 3D scenes is lighting. The light role is responsible for more than just controlling basic intensity, contrast, or color parameters. Lights are interactive and can create different effects, from controlling global illumination for fade and dim effects, to partial illumination casting shadows, to color toning and shading individual objects. Such effects can be created via an intensity slider and color picker. Figure 8 shows the first AR designer embodying ambient light to increase brightness and fade in the Lion King opening scene.

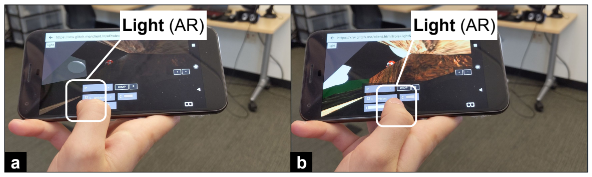

Figure 8. Light increases the brightness in Lion King's opening scene.

Controlling the Camera

XRDirector supports natural camera movements known from filmmaking: *pan* by rotating the device while keeping the position fixed to change the orientation; *dolly* by moving the device to change the position in the virtual world in the AR/VR modes; and *zoom* by moving a slider to magnify the scene without changing the camera position. Figure 9 shows the second AR designer in the camera role moving the camera (here, the AR capable phone) to zoom in on the Rafiki character.

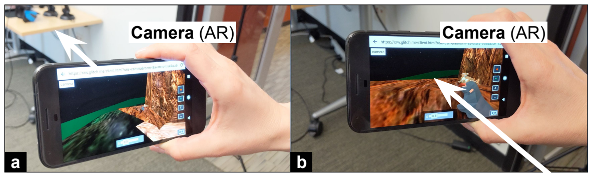

Figure 9. Camera moves forward (dolly-in) to zoom on Rafiki in AR.

Controlling 3D Characters

Similar to lights and the camera, designers in the actor role can control 3D characters by using XRDirector to translate AR/VR device motions in physical space to object motions in virtual space. Because AR/VR device can be tracked along six degrees-of-freedom at fairly high precision, this gives designers full control over character animation and allows them to freely express interactions. The character's appearance can be changed using widgets for the 3D model, color, size, opacity, and visibility. Figure 10 shows the third designer in the actor role controlling the Rafiki character to simulate walking up Pride Rock.

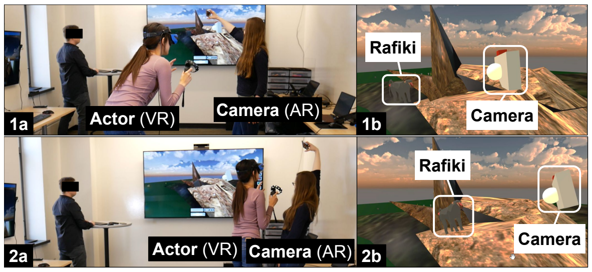

Figure 10. Actor moves up Pride Rock in VR while camera moves backward (dolly-out) in AR mode to keep the zoom on Rafiki the same.

Rescaling the Virtual World

Figure 10 also shows the second designer moving the camera to keep the zoom on the Rafiki character the same during movement. This requires both time-based and spatial coordination between VR actor and the AR camera. In this paper, we focus primarily on spatial coordination between AR/VR designers and aim to show that the director role is vital to address these issues. Figure 10 is representative of two of the issues we will detail later: *spatial constraints* and *spatial mappings*. One of the special functions the director has is the rescale function. Figure 11 shows the director temporarily reducing the scale of the virtual world such that the designer's physical movements translate to larger object movements in the scene. Additionally, the AR actor could choose a different starting position for the camera and recenter the virtual world. This would lead to different virtual–physical spatial mappings between the AR and VR actors. As a result, the AR camera would see the VR actor at a different virtual position than where they are in physical space, but this would make better use of limited physical space and avoid collisions.

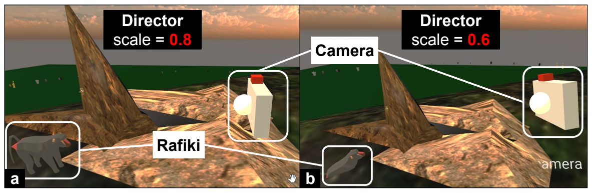

Figure 11. Rescaling the scene to make do with limited physical space.

Record & Replay of Object Manipulations

Since changes made to any virtual object take immediate effect in the scene, the same tools used for authoring can be used for performance animation and Wizard of Oz.

Recording performances

Designers in the camera or director roles have access to XRDirector's record & replay functions to create animations and effects (Figure 7c and d). When recording is initiated, XRDirector captures the states of the 3D characters, lights, and cameras in terms of their position, rotation, and size. Recorded object trajectories are visualized via motion paths (Figure 7c). Our implementation also keeps track of a set of pre-specified components (material color and opacity, light color and intensity, and camera zoom, per default) in real-time. After recording these initial states, XRDirector continuously tracks changes to these properties and builds a timeline of events. This makes it possible to record natural 3D character movements, cross-fading light effects, changing brightness or color, and camera transitions. Similar to Dontcheva et al. [8], our record method allows iterative, overlapping recordings in which additional properties can be modulated along a shared timeline. This enables detailed and concurrent animations of more than one object even by a single designer in the director role.

Synchronized playback across devices

XRDirector also features a replay tool that reproduces the recorded animation steps and synchronizes them with other devices. To do this, our replay tool first hides all current puppets in the scene for all clients, then spawns ghost actors controlled by the system for each of the recorded puppets that recreate the recorded states in real-time. The actual animation script therefore only needs to be invoked for the designer in the camera/director role, while the ghost actors ensure state-based client synchronization. This networked replay approach is lightweight and happens in real-time (avoiding transfer of the entire animation script with all its properties). It also allows other clients to join a session even during record & replay and still be in sync with all other clients.

Post-processing of Recorded Animations

To enable the director to adjust recorded animations post-hoc, we developed two additional interfaces to scale movement trajectories and animation timelines.

Rescaling movement trajectories

We implemented a proof-of-concept interface for adjusting trajectories using input fields to modify the scale along x, y, and z dimensions independently. Figure 12 shows an example where we downscaled the recorded trajectory for the Rafiki character to better match the scale of our Pride Rock 3D model and reproduce the movement from the Lion King teaser. Our algorithm determines the geometric center as the point around which the animation will be rescaled, then applies vector transformations for all recorded trajectory points.

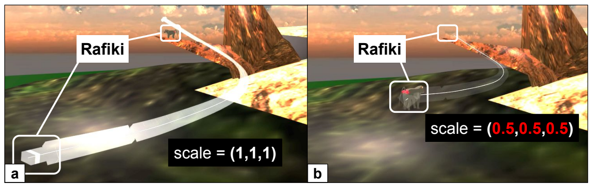

Figure 12. Downscaling the recorded trajectory of the Rafiki character.

Layering animation timelines

We also implemented a interface for freely adjusting the timing of animations recorded for one or more objects. The director can adjust the start and end times of each animation by dragging the handles and uncheck animations to disable them. This adjusts the delay and duration property of the corresponding A-Frame animation component. The interface therefore allows to correct animations created by multiple designers where timing was not right or to try out and toggle alternative animations. It also enables more complex animations based on multiple separate demonstrations of even a single designer to be layered [8]. Figure 13 shows our proof-of-concept layering of two animation sequences for the Rafiki character.

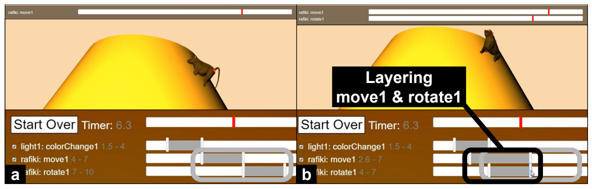

Figure 13. Layering move & rotate animation sequences of Rafiki.

IMPLEMENTATION

XRDirector is implemented using the web-based A-Frame XR development platform originally created by Mozilla. While A-Frame XR supports a rapidly growing range of mixed-reality devices, we have successfully tested the system on Android phones with ARCore on Google's WebARonARCore browser, Windows Mixed Reality with SteamVR on Firefox, and HoloLens on Edge. A new AR/VR device can start or join an existing prototyping session simply by navigating to a shared URL in a WebXR capable browser. Real-time synchronization between multiple users and devices is implemented using Networked A-Frame based on the WebRTC standard using the easyrtc library. Our record & replay tools are implemented using the A-Frame animation component which is based on the anime.js animation engine.

FINAL WORKSHOPS WITH XRDIRECTOR

As a formative evaluation of our system, we conducted follow-up workshops with the same 12 students we previously recruited, asking them to use XRDirector to recreate scenes they found challenging to build in A-Frame and Unity.

Method

We asked participants to create four teams of three each. From the 12 scenes discussed in the initial workshops, we selected four with at least 2–3 animated 3D characters, light effects and camera movements, then assigned the director role to the original creator of each selected scene and assigned actor roles to the remaining two team members. After a brief introduction to our system, teams were given 90 minutes to practice their roles and recreate the 2–3 animations with XRDirector. We set up three work stations running XRDirector per team, one station with a 48" screen, one with a Windows Mixed Reality VR headset and controllers, and one with an ARCore enabled smartphone. Each team also had a neutral observer from the research team that recorded prototyping activity and answered technical questions. Teams stayed the same and worked separately but came together in a discussion round. Finally, participants filled in a feedback questionnaire on strengths and weaknesses of XRDirector,and received a $20 gift card as compensation for their time.

Results

Below, we present our analysis of participant feedback and observer notes using thematic coding.

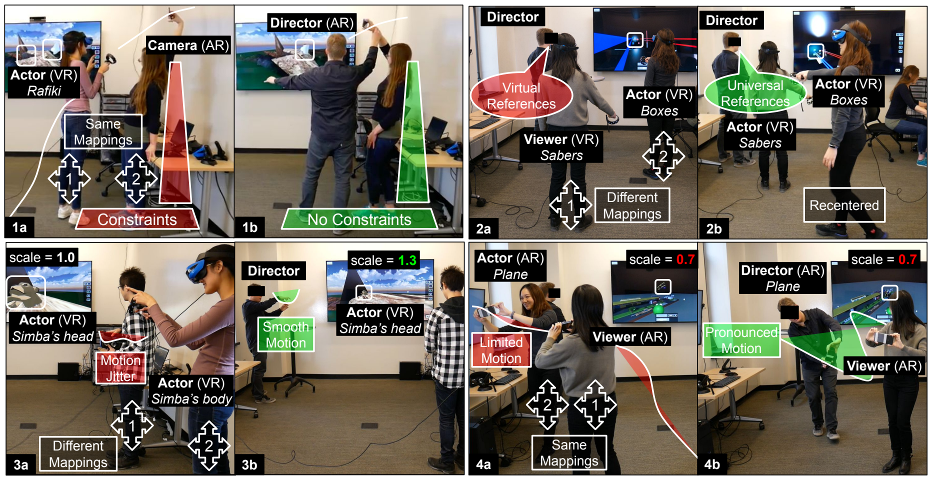

Figure 14. Final case studies with XRDirector: ***(1a)*** **spatial coordination between VR actor and AR camera was hard due to the same mappings resulting in physical constraints;** ***(1b)*** **director kept the mappings but demonstrated how to optimize camera movement with limited physical space;** ***(2a)*** **spatial coordination between VR actor and VR viewer was hard due to different mappings and virtual references given by director not matching the physical world ("move towards [the VR viewer]");** ***(2b)*** **director kept the mappings but moved the VR actor to a matching physical location and used universal references ("move back and forth");** ***(3a)*** **spatial coordination was hard for two VR actors who wanted to coordinate small movements of a character's head and body due to low resolution of the scene, resulting in jitter and uncontrolled motions of the tiny character parts;** ***(3b)*** **director upscaled the scene, which relaxed character movement, and demonstrated how to coordinate motions between the VR actors given the different mappings;** ***(4a)*** **spatial coordination between AR actor and AR viewer was hard due to low resolution resulting from the downscaled scene to use the same mappings, which limited character movement;** ***(4b)*** **director demonstrated how to pronounce physical motion to increase the realism of character movement.**

Animation without coding is a strong benefit

In the initial workshops, the participants expressed difficulty creating sequential animations through scripting. They rated the experience using XRDirector to create animations as being easy and helpful, especially for AR actors: "Easy to create animation curves." "It is really helpful for prototyping animation." "No need to write code! That is nice!" The ability to watch the replay synchronized on all devices was also highlighted: "The animation replication feature is super cool."

Flexible tool for collaboration in mixed reality

While we observed an initial learning curve, participants had no trouble understanding their roles: "Users can easily understand the role and start testing this system." The role of the director was critical to the success of the collaboration activity. They guided the actors to complete their tasks through hand gestures, verbal directions, and physically moving the actors into position. When directors did not give sufficient directions, actors would look at other actors and the director's screen to orient themselves in the scene. Complex coordination between actors was still difficult to achieve, especially when one actor was in VR and the other in AR. Because participants started from scratch and recreated the 3D characters, the actors did not have any landmarks, making coordination harder [25].

All roles are important

Having the director change perspective and take control of actor actions solved confusion in several cases. For example, when the actors had difficulty scaling their 3D character or knowing when their avatar was dressed with the selected object, the director stepped in. In one team, the director flew above the actors to gain a greater perspective on the scene and provide more accurate verbal directions for the actors. For the actors, however, this strategy introduced new coordination challenges because the director's avatar was not in their field of view anymore. This showed us the importance of the camera as common reference point for actors.

FINAL CASE STUDIES WITH XRDIRECTOR

In this section, we revisit our three case studies and report on our experiences recreating them with XRDirector.

We selected a total of six scenes from our three case studies. One of the authors assumed the director role and briefly recreated every scene with a special producer mode of XRDirector simulating up to four roles on the same device side-by-side (cf. Figure 7). This allowed him to develop a plan for instructing the roles and suggest modes when the research team came together for the reenactments summarized below. While we had access to more devices, we ended up needing two VR setups, two AR phones, and one large screen for the director. We recorded our final case studies for video analysis.

Figure 14 shows examples from our recreated scenes and illustrates common challenges. We start by analyzing the required effort, then discuss common challenges and how we addressed them using XRDirector, and conclude with a comparison of the strengths and weaknesses we noticed with different role/mode combinations.

We were able to recreate the six selected scenes from our physical and digital prototypes using collaborative immersive authoring with XRDirector. Since the quality of the created product depends on time, we fixed the times prior to the experiment. Including filming and fixing some technical issues with SteamVR, we spent three hours to recreate four scenes from Lion King and one hour for Beat Saber and ARcade Plane each. Compared to the low effort for the physical prototypes and high effort for the digital A-Frame and Unity prototypes (with high-fidelity 3D models and scripted animations closely following the original character, light, and camera movement), we found the effort with XRDirector to be low–medium with major time savings due to collaborative animations production.

Spatial Coordination Challenges

In our video analysis, we identified issues related to spatial coordination and selected representative examples shown in Figure 14. We found four classes of issues:

- **spatial awareness**: differences in designers' positions relative to virtual or physical objects, differences between virtual and physical world geometry, or differences in the designers' views with no common reference points cause coordination challenges—most challenging for the director instructing actors on where and how to move in virtual or physical space, for example, when the director wanted the AR camera or VR actor to move in a particular way or to a specific location (Figure 14.1 and 3).
- **spatial constraints**: differences between dimensions of the virtual and the physical worlds or differences between volumetric shapes of the virtual and physical world cause coordination challenges—most challenging when physical space is too limited, for example, when the VR actor and AR camera almost collided (Figure 14.1), or when physical geometry makes it hard to perform movements that match the requirements of the virtual world, for example, when Rafiki was to move up Pride Rock the actor simulated this by initially bending down (Figure 9.1);
- **spatial mappings**: differences in coordinate spaces between devices or accumulated tracking error cause coordination challenges—most challenging when designers use the same mode, for example, when one VR actor was to move the cubes while the VR player was swinging the sabers (Figure 14.2) or when one AR actor was to fly the plane while the AR camera was to pan and tilt to keep it in view (Figure 14.4);
- **spatial resolutions**: differences in tracking precision between devices or due to rescaling the virtual world cause coordination challenges—most challenging when actors needed to perform slow or no movements, for example, when Simba's head was to turn while keeping the rest of the body still and two VR actors simulated this with one turning the HMD controlling the head and the other keeping the body pose still (Figure 14.3), or when movements had to be compressed/exaggerated to counterbalance reduced/increased world scale, for example, when the plane was to fly realistically and the AR actor had to perform almost parabolic motion (Figure 14.4);

Workarounds in XRDirector and Trade-offs

We note that most instances of spatial coordination challenges were of more than one class of issue and that the challenges seemed harder the more issues came together. Our video analysis showed that the functions provided by XRDirector helped address the vast majority of issues, but that selection and use of the best function to counteract an issue required practice and making trade-offs. The two key functions we made extensive use of were the recenter and rescale functions.

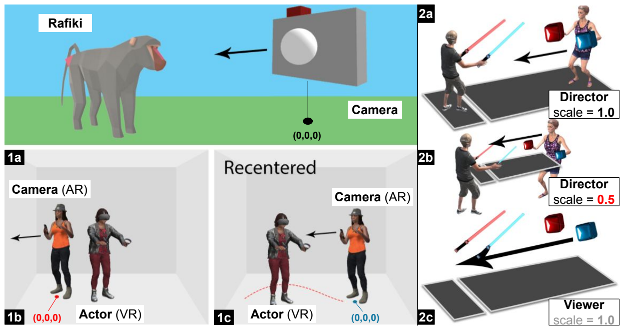

Figure 15. Recenter and rescale functions: *1)* in Lion King, the camera moves to a different spot in the room and recenters the virtual world (reset position to $(0,0,0)$) to make better use of limited physical space in AR; *2)* in Beat Saber, the director rescales the virtual world (reduce scale to 0.5) to allow the actor moving the cubes to travel farther in VR—the viewer is not impacted by this and views everything at original scale.

Recenter Function

Workarounds. Figure 15.1 illustrates the use of the recenter function to resolve spatial constraints and align spatial mappings between designers. To work with limited physical space, the AR camera first moved to a different physical location, then recentered the virtual world to reset their location in the virtual world and demonstrate the movement starting from that position. When done as illustrated, this also aligned the spatial mapping with the location of the VR actor so that the designer in the camera role could see the actor and Rafiki character at matching positions in AR.

Trade-offs. Figure 14.1 depicts a near collision due to limited physical space for the subsequent dolly-out move. The AR camera could have moved to a different physical position and could have recentered again. The trade-offs are that this would require a cut and a second take to continue the scene and would again increase the differences in spatial mappings between designers. Therefore, the director resolved the issue by downscaling the world and guiding the AR camera's hand movement to record the dolly-in and out in one take.

Rescale Function

Workarounds. Figure 15.2 illustrates the use of the rescale function to increase spatial resolution and take advantage of different spatial mappings. One of the primary challenges in Beat Saber was that the cubes had to move increasingly faster towards the player with the sabers over a relatively long distance in virtual space. The director solved this by downsizing the virtual world such that the virtual objects shrink in size. As a result, the same physical motion as at the original size would make the cubes appear to move faster and farther in virtual space. Importantly, the change in world scale is not visible to viewers, resulting in a seamless viewer experience.

Trade-offs. The difference in perceived motion when changing the size of the virtual world creates a mismatch between the designer's movement in physical space and the resulting 3D character's trajectory. For example, when the virtual world is scaled very large, dolly moves in virtual space will need to be exaggerated, while tilts and pans directly match the designer's movement in physical space. While we were able to balance these differences with practice, they could also be

Modes / Roles	3D mode		AR mode		VR mode	
	Good at	Bad at	Good at	Bad at	Good at	Bad at
Director	instruct AR/VR designers relative to physical/virtual position	know how scene views in AR/VR	instruct AR designers relative to physical/virtual position	instruct VR designers if their real/virtual positions don't match	instruct VR designers relative to virtual position	instruct AR designers if their real/virtual positions don't match
Actors	plan movement of 3D character	embody 3D character act movement out naturally	plan movement of 3D character act movement out naturally	understand virtual spatial references	embody 3D character act movement out naturally	understand physical spatial references
Lights	control lights/shadows precisely	know how lights/shadows view in AR/VR	control lights/shadows on physical objects	know how lights/shadows view in VR	control lights/shadows on virtual objects	know how lights/shadows view in AR
Camera	move camera precisely pan camera along one axis	move camera naturally know how scene views in AR/VR	move camera naturally	pan camera along one axis know how scene views in 3D/VR	move camera naturally know how scene views in 3D	pan camera along one axis know how scene views in AR

Figure 16. Roles and modes in XRDirector: comparing strengths and weaknesses of each role/mode combination.

addressed algorithmically by adding support for non-linear mappings in XRDirector via gain functions [30].

Strengths and Weaknesses of Role/Mode Combinations

We found that XRDirector's expressive power and flexibility comes from the flexibility to match roles and modes. The best role/mode match is a function of *i)* the type of virtual object being controlled, *ii)* where the object is located in physical space, *iii)* how the object needs to move in virtual space.

Using XRDirector to recreate the case studies and our test-driven development of each role and mode allowed us to identify strengths and weaknesses of role/mode combinations. Figure 16 details our analysis. The overall concerns directly relate to the identified spatial challenges: knowing how the scene views for other designers, having precise control over movements, and acting on the director's instructions.

In terms of overall preferences, our designers in the actor role preferred to animate a 3D character according to their exact head movements in VR mode, designers in the camera role preferred the AR mode on the phone, in both cases because it seemed more intuitive to them. Our director preferred the 3D mode to maintain an overview of what happens in the virtual world and because he did not interfere with designers performing in physical space.

ADDITIONAL APPLICATIONS

In this section, we present two additional applications we created with XRDirector (Figure 17). These examples are meant to demonstrate creative usage of XRDirector and ways of enabling Wizard of Oz. They are also interesting because they started to look at possible future extensions.

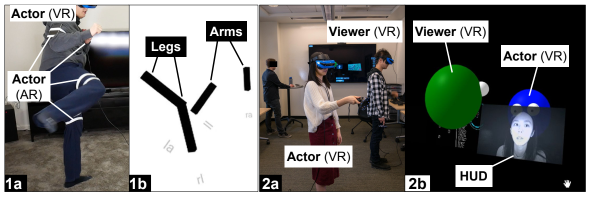

Figure 17. More applications: *1)* Kung Fu Stickman based on four AR phones strapped to the VR actor's limbs; *2)* Iron Man HUD simulated by a VR actor placing a video screen to face a VR viewer.

Kung Fu Stickmen

Our first application was a recreation of the once popular Kung Fu Stickman series (Figure 17.1). The series consisted of stick figure fighting scenes animated in Flash. Our idea was to instrument a VR actor with multiple AR phones to track his joints and animate a stick figure. The first attempt was to create the puppet by nesting a sphere for the head and stretched boxes for the arms and legs in A-Frame. This worked in principle to animate the different joints. However, the lack of a precise calibration method for ARCore Android phones and the fact that tracking error accumulates quickly with rapid movements caused the joints not to hold together nicely. Our solution was to add physics to the scene and add joints with cone twist constraints to the stickfigure. This then caused the issue that physics would be run and simulated on each device, for which we developed a method that only the director runs the physics to animate dynamic bodies such that body movement is automatically synced with other clients.

Iron Man

Our second application was a recreation of the Iron Man HUD (Figure 17.2). Our idea was to allow a viewer to experience the HUD on a VR headset, while a VR actor moves different screens into view to compose the HUD for the viewer. We prototyped the HUD screens as an animated slidedeck consisting of several layers in Keynote and exported each layer as a separate video. While it was fast and easy to compose the A-Frame scene from the video assets mapped onto semi-transparent planes, it was initially hard for the VR actor to control visibility and simulate a dynamic HUD for the viewer. Our solution was to use black fog in the scene around the viewer so that different HUD components could be quickly and easily shown or hidden just based on distance from the viewer. As an extension of this, we built on XRDirector's WebRTC infrastructure to render a life video feed from the actor on one device to the headset, enabling the simulation of a video chat or personal assistant by another actor.

CONCLUSION

We presented *XRDirector*, a collaborative mixed-reality system that orchestrates multiple designers working on AR/VR authoring tasks by assigning them specific roles inspired from film making. We showed that XRDirector is able to recreate fairly complex 3D, VR, and AR scenes in less than 30 minutes, which previously required hours of AR/VR development effort. Based on our investigations around XRDirector, we see a lot of generative power and a flexible platform for experimentation. While we included initial usages of the system for Wizard of Oz, we believe that this is one of the main future research directions. We hope that XRDirector's tight integration with A-Frame can reduce the learning curve and enable progression towards fuller usage of such AR/VR development platforms. We observed how our role-based prototyping framework makes key concepts of mixed-reality interfaces such as virtual cameras, 3D animation, and lighting easily understandable. We hope that tools like XRDirector will enable broader participation of less technical designers. This is something we would like to enable by making XRDirector available to interested researchers and practitioners.

Acknowledgments

Thanks to Ilma Bilic and Amy (Shih-Ting) Lin for their help with the described prototypes and reenactments in XRDirector.

REFERENCES

[1] Raphael Anderegg, Loïc Ciccone, and Robert W. Sumner. 2018. PuppetPhone: puppeteering virtual characters using a smartphone. In *Proc. MIG*. 5:1–5:6.

[2] Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B. Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. 2008. Video puppetry: a performative interface for cutout animation. *TOG* 27, 5 (2008), 124:1–124:9.

[3] Mark Billinghurst and Hirokazu Kato. 1999. Collaborative mixed reality. In *Proc. ISMAR*. 261–284.

[4] Lung-Pan Cheng, Patrick Lühne, Pedro Lopes, Christoph Sterz, and Patrick Baudisch. 2014. Haptic turk: a motion platform based on people. In *Proc. CHI*. DOI: http://dx.doi.org/10.1145/2556288.2557101

[5] Lung-Pan Cheng, Sebastian Marwecki, and Patrick Baudisch. 2017. Mutual Human Actuation. In *Proc. UIST*. 797–805.

[6] Lung-Pan Cheng, Thijs Roumen, Hannes Rantzsch, Sven Köhler, Patrick Schmidt, Robert Kovacs, Johannes Jasper, Jonas Kemper, and Patrick Baudisch. 2015. TurkDeck: Physical Virtual Reality Based on People. In *Proc. UIST*. 417–426.

[7] Richard C. Davis, Brien Colwell, and James A. Landay. 2008. K-sketch: a 'kinetic' sketch pad for novice animators. In *Proc. CHI*. 413–422.

[8] Mira Dontcheva, Gary D. Yngve, and Zoran Popovic. 2003. Layered acting for character animation. *TOG* 22, 3 (2003), 409–416. DOI: http://dx.doi.org/10.1145/882262.882285

[9] Steven Dow, Jaemin Lee, Christopher Oezbek, Blair MacIntyre, Jay David Bolter, and Maribeth Gandy. 2005a. Wizard of Oz interfaces for mixed reality applications. In *Proc. CHI Extended Abstracts*. 1339–1342.

[10] Steven Dow, Blair MacIntyre, Jaemin Lee, Christopher Oezbek, Jay David Bolter, and Maribeth Gandy. 2005b. Wizard of Oz support throughout an iterative design process. *IEEE Pervasive Computing* 4, 4 (2005), 18–26.

[11] Barrett Ens, Joel Lanir, Anthony Tang, Scott Bateman, Gun Lee, Thammathip Piumsomboon, and Mark Billinghurst. 2019. Revisiting collaboration through mixed reality: The evolution of groupware. *IJHCS* 131 (2019).

[12] Jamie Feltham. 2017. Disney Is Using VR To Help Film The Lion King Remake. Available from uploadVR at https://uploadvr.com/disney-using-vr-film-lion-king-remake/. (July 2017).

[13] Maribeth Gandy and Blair MacIntyre. 2014. Designer's augmented reality toolkit, ten years later: implications for new media authoring tools. In *Proc. UIST*.

[14] Taejin Ha, Woontack Woo, Youngho Lee, Junhun Lee, Jeha Ryu, Hankyun Choi, and Kwanheng Lee. 2010. ARtalet: tangible user interface based immersive augmented reality authoring tool for Digilog book. In *Proc. ISUVR*. 40–43.

[15] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D puppetry: a kinect-based interface for 3D animation.. In *Proc. UIST*. 423–434.

[16] Andrew J. Hunsucker, Kelly McClinton, Jennifer Wang, and Erik Stolterman. 2017. Augmented Reality Prototyping For Interaction Design Students. In *Proc. CHI Extended Abstracts*. 1018–1023.

[17] Annie Kelly, R. Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. 2018. ARcadia: A Rapid Prototyping Platform for Real-time Tangible Interfaces. In *Proc. CHI*.

[18] Boriana Koleva, Ian Taylor, Steve Benford, Mike Fraser, Chris Greenhalgh, Holger Schnädelbach, Dirk vom Lehn, Christian Heath, Ju Row-Farr, and Matt Adams. 2001. Orchestrating a mixed reality performance. In *Proc. CHI*. 38–45.

[19] Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. 2015. Apparition: Crowdsourced User Interfaces that Come to Life as You Sketch Them. In *Proc. CHI*.

[20] Gun A. Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. 2004. Immersive Authoring of Tangible Augmented Reality Applications. In *Proc. ISMAR*. 172–181.

[21] Sang Won Lee, Yujin Zhang, Isabelle Wong, Yi Wei Yang, Stephanie D. O'Keefe, and Walter S. Lasecki. 2017. SketchExpress: Remixing Animations for More Effective Crowd-Powered Prototyping of Interactive Interfaces. In *Proc. UIST*. 817–828.

[22] Germán Leiva and Michel Beaudouin-Lafon. 2018. Montage: A Video Prototyping System to Reduce Re-Shooting and Increase Re-Usability. In *Proc. UIST*. 675–682.

[23] David Lindlbauer and Andy D. Wilson. 2018. Remixed Reality: Manipulating Space and Time in Augmented Reality. In *Proc. CHI*. 129.

[24] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proc. UIST*.

[25] Jens Müller, Roman Rädle, and Harald Reiterer. 2016. Virtual Objects as Spatial Cues in Collaborative Mixed Reality Environments: How They Shape Communication Behavior and User Task Load. In *Proc. CHI*. 1245–1249.

[26] Michael Nebeling and Katy Madier. 2019. 360proto: Making Interactive Virtual Reality Augmented Reality Prototypes from Paper. In *Proc. CHI*.

[27] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications. In *Proc. CHI*.

[28] Michael Nebeling and Maximilian Speicher. 2018. The Trouble with Augmented Reality/Virtuality Reality Authoring Tools. In *Proc. ISMAR Adjunct*.

[29] Michael Nitsche and Sanjeev Nayak. 2012. Cell Phone Puppets: Turning Mobile Phones into Performing Objects. In *Proc. ICEC*. 363–372.

[30] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proc. UIST*. 79–80.

[31] Dieter Schmalstieg, Anton Fuhrmann, Gerd Hesina, Zsolt Szalavári, L Miguel Encarnaçao, Michael Gervautz, and Werner Purgathofer. 2002. The studierstube augmented reality project. *Presence: Teleoperators & Virtual Environments* 11, 1 (2002), 33–54.

[32] Harrison Jesse Smith and Michael Neff. 2018. Communication Behavior in Embodied Virtual Reality. In *Proc. CHI*. 289.

[33] Maximilian Speicher, Brian D. Hall, and Michael Nebeling. 2019. What is Mixed Reality?. In *Proc. CHI*.

[34] Maximilian Speicher and Michael Nebeling. 2018. GestureWiz: A Human-Powered Gesture Design Environment for User Interface Prototypes. In *Proc. CHI*.

[35] Jia Wang, Owen Leach, and Robert W. Lindeman. 2013. DIY World Builder: An immersive level-editing system. In *Proc. 3DUI*. 195–196.